# 开放式机器学习课程。专题3。分类决策树与k近邻



你好！这是本系列的第三篇文章。今天我们终于达到了机器学习。这将是令人兴奋的！

### 文章概要

1. 介绍
2. 决策树

* 如何建立决策树
* 树生成算法
* 分类问题中拆分的其他质量标准
* 决策树如何处理数字特征
* 关键树参数
* Scikit learn中的班级决策者
* 回归问题中的决策树

1. 最近邻法

* 近邻法在实际应用中的应用
* Scikit-learn中的kneighborsscrifier类

1. 模型参数选择与交叉验证
2. 应用实例及复杂案例

* 客户流失预测中的决策树和最近邻法
* 决策树的复杂情况
* 决策树与k-NN在MNIST手写数字识别中的应用
* 近邻法的复情形

1. 决策树与最近邻法的优缺点
2. 作业#3
3. 有用资源

如果您克隆了，则以下材料最好视为A，并且可以使用Jupyter在本地复制。

### 一。介绍

在我们深入阅读本周文章的材料之前，让我们先来谈谈我们将要解决的问题，以及它在令人兴奋的机器学习领域中的地位。T.Mitchell的《机器学习》（1997）一书给出了机器学习的经典、一般定义如下：

一个计算机程序被称为从经验E中学习某些类别的任务T和性能度量P，如果它在T中的任务（由P度量）的性能随着经验E而提高。

在各种问题设置中，T、P和E可以指完全不同的东西。机器学习中最受欢迎的任务有：

* 根据实例的特征将其分类为某一类别；
* 基于实例其他特征的数值目标特征的回归预测；
* 聚类 - 根据实例的特征识别实例的分区，使组内的成员比其他组中的成员更相似；
* 异常检测 - 搜索与其余样本或某组实例“极不相似”的实例；
* 还有更多。

《机器学习基础》一章提供了一个很好的概述（作者：Ian Goodfelle，Yoshua Bengio，Aaron Courville，2016）。

经验E指的是数据（没有它我们哪儿也去不了）。机器学习算法可以分为有监督和无监督两种。在无监督学习任务中，有一组由一组特征描述的实例组成。在监督学习问题中，还有一个目标变量，这是我们希望能够预测的，在训练集中的每个例子中都是已知的。

#### 例子

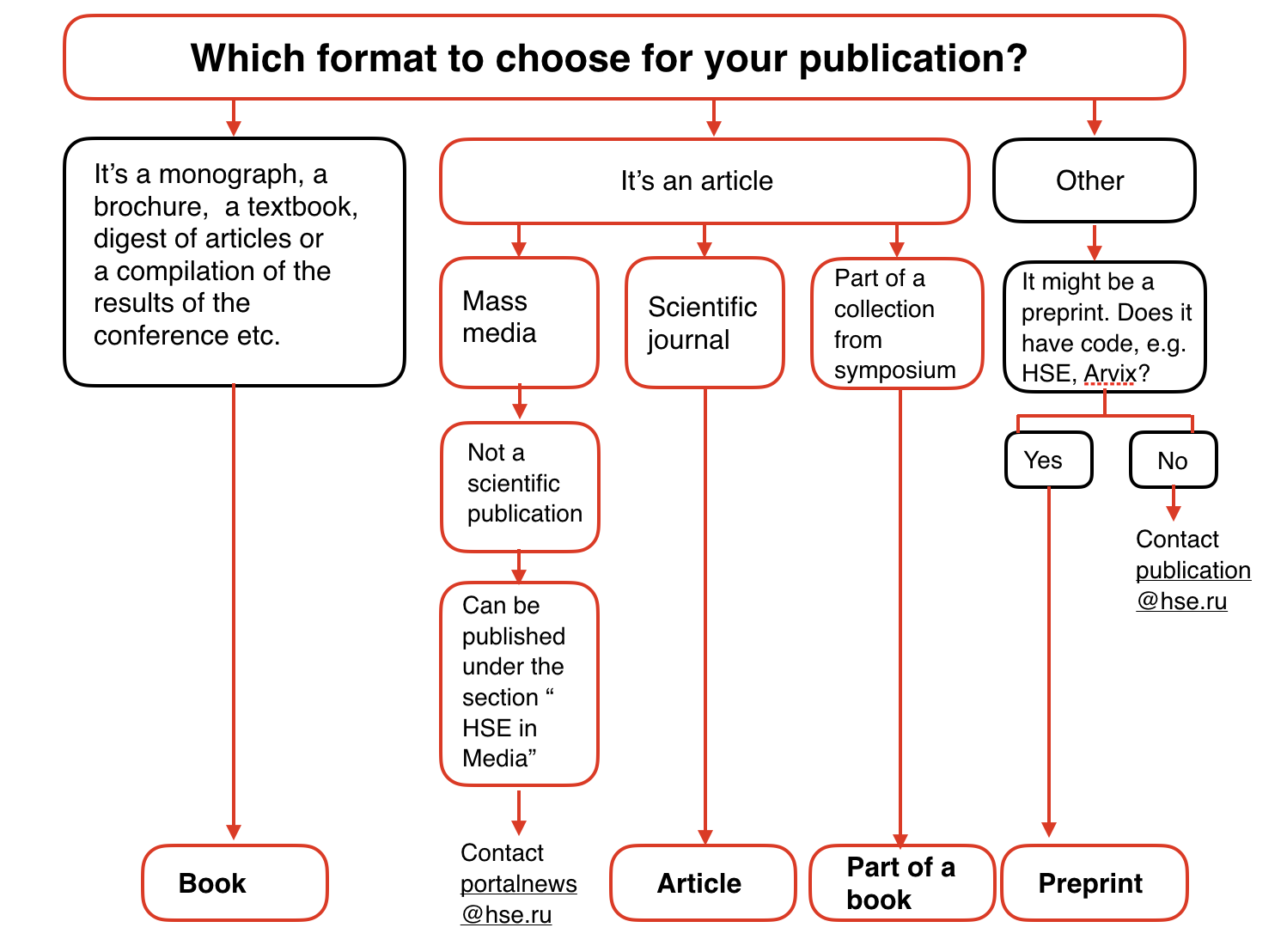
分类和回归是有监督的学习问题。例如，作为一家信贷机构，我们可能希望根据积累的客户数据预测贷款违约。在这里，经验E是可用的培训数据：一组实例（客户）、每个实例的特征集合（例如年龄、工资、贷款类型、过去的贷款违约情况等）和一个目标变量（它们是否在贷款中违约）。这个目标变量只是贷款违约（1或0）的一个事实，所以回想一下，这是一个（二进制）分类问题。如果你用贷款逾期的时间来预测，这将成为一个回归问题。

最后，机器学习定义中使用的第三个术语是算法性能评估P的度量。这些度量因各种问题和算法而异，我们将在研究新算法时对它们进行讨论。现在，我们将引用一个简单的分类算法度量，即测试集上正确答案的比例 - 准确性 - 。

让我们来看看两个有监督的学习问题：分类和回归。

### 2。决策树

我们从最流行的分类和回归方法之一决策树开始概述分类和回归方法。决策树不仅用于机器学习，还用于日常生活中的决策。流程图实际上是决策树的可视化表示。例如，高等经济学院发布信息图表，让员工的生活更轻松。以下是在机构门户网站上发布论文的部分说明。



就机器学习而言，可以将其视为一个简单的分类器，它根据内容（书籍、小册子、论文）、期刊类型、原始出版物类型（科学期刊，程序）等。

决策树通常是专家经验的概括，是共享特定过程知识的一种手段。例如，在引入可伸缩机器学习算法之前，银行业的信用评分任务由专家解决。贷款的决定是基于一些直觉（或经验）衍生的规则做出的，这些规则可以表示为一个决策树。

在我们的下一个例子中，我们解决了基于“年龄”、“房屋所有权”、“收入”和“教育”的二元分类问题（批准/拒绝贷款）。

决策树作为一种机器学习算法，本质上与上图相同；我们将“特征a值小于x，特征b值小于y…”=>类别1“形式的逻辑规则流合并到树状数据结构中。这种算法的优点是易于解释。例如，利用上述方案，银行可以向客户解释为什么拒绝贷款：例如，客户没有住房，收入不足5000英镑。

正如我们稍后将看到的，许多其他模型虽然更精确，但不具有此属性，可以被视为更多的“黑箱”方法，在这种方法中，很难解释如何将输入数据转换为输出数据。由于这种“可理解性”和与人类决策的相似性（您可以很容易地向您的老板解释您的模型），决策树获得了极大的普及。C4.5是这一类分类方法的代表，甚至是十大最佳数据挖掘算法中的第一个（“数据挖掘十大算法”，知识和信息系统，2008年）.

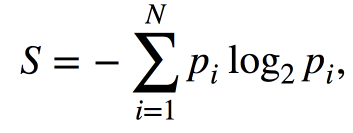
### 如何建立决策树

早些时候，我们看到发放贷款的决定是基于年龄、资产、收入和其他变量。但首先要看什么变量呢？让我们讨论一个简单的例子，其中所有的变量都是二进制的。

回想一下“20个问题”的游戏，在引入决策树时经常会提到这个游戏。你可能玩过这个游戏，一个人想到名人，而另一个人只问“是”或“否”的问题来猜测。猜者会先问什么问题？当然，他们会问一个缩小了剩余选择的数量最多。问“是安吉丽娜·朱莉吗？”？如果是负面回应，除了一个名人之外，其他人都有可能成为名人。相反，问“名人是女人吗？”？“将把可能性降低到大约一半。这就是说，“性别”特征比“安吉丽娜·朱莉”、“西班牙人”或“热爱足球”等其他特征更好地分离了名人数据集，这与基于熵的信息获取概念相对应。

#### 熵

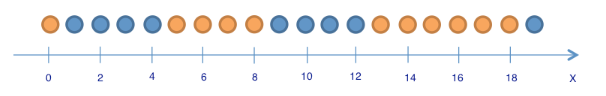
香农熵定义为一个具有N种可能状态的系统，如下所示：



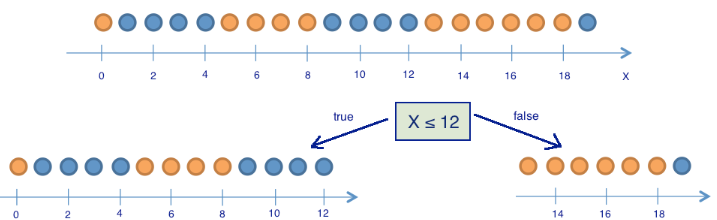
其中Pi是在第i状态下找到系统的概率。这是一个非常重要的概念，应用于物理学、信息论等领域。熵可以用系统的混沌程度来描述。熵越高，系统的有序性越低，反之亦然。这将有助于我们正式确定“有效的数据分割”，我们在“20个问题”中提到了这一点。

#### 玩具示例

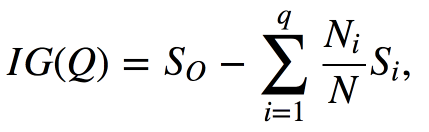
为了说明熵如何帮助我们识别用于构建决策树的良好特性，让我们看一个玩具示例。我们将根据球的位置来预测它的颜色。



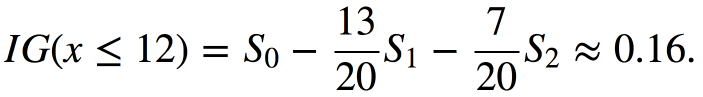
有9个蓝球和11个黄球。如果我们随机抽出一个球，那么它是蓝色的概率p1=9/20，黄色的概率p2=11/20，这给了我们一个熵S0=-9/20log2（9/20）-11/20log2（11/20）≈1。这个值本身并不能告诉我们多少，但是让我们看看如果我们把球分成两组，这个值是如何变化的：位置小于或等于12，大于12。



左边组有13个球，8个蓝色，5个黄色。该群的熵为S1=-5/13log2（5/13）-8/13log2（8/13）≈0.96。右边组有7个球，1个蓝色，6个黄色。右群的熵为S2=-1/7log2（1/7）-6/7log2（6/7）≈0.6。正如你所看到的，熵在两组中都降低了，在正确的组中更是如此。由于熵实际上是系统中的混沌（或不确定性）程度，因此熵的减少称为信息增益。形式上，基于变量Q（在本例中为变量“x≤12”）的拆分的信息增益（IG）定义为



其中q是分割后的组数，Ni是样本中变量q等于第i个值的对象数。在我们的例子中，我们的分裂产生了两组（q=2），一组有13个元素（N1=13），另一组有7个元素（N2=7）。因此，我们可以将信息增益计算为



结果发现，用“坐标小于等于12”的方法把球分成两组，我们得到了一个更有序的系统。让我们继续把它们分成几组，直到每组中的球都是相同的颜色。

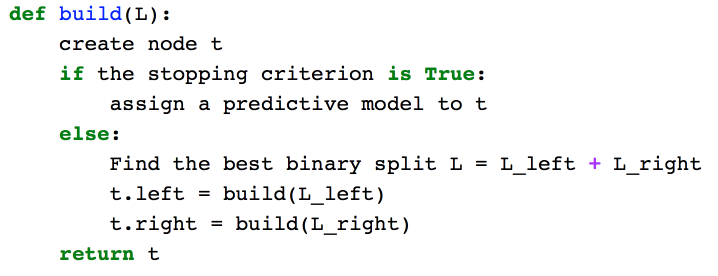
对于正确的组，我们可以很容易地看到，我们只需要一个额外的分区使用“坐标小于或等于18”。但是，对于左派，我们还需要三个。注意，所有球颜色相同的组的熵等于0（log2（1）=0）。

我们成功地构建了一个基于球的位置预测球颜色的决策树。如果我们添加任何球，这个决策树可能不会很好地工作，因为它完全适合训练集（最初的20个球）。如果我们想在这种情况下做得好，一棵“问题”更少或分裂更少的树将更准确，即使它不完全适合训练集。我们稍后将讨论过拟合的问题。

### 树生成算法

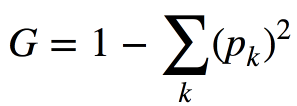
我们可以确保在前面的例子中构建的树是最优的：只需要5个“问题”（以变量x为条件）就可以将决策树完美地匹配到训练集在其他拆分条件下，生成的树会更深，即采取更多的“问题”来获得答案。

在决策树构造的流行算法（ID3或C4.5）的核心在于信息增益贪婪最大化原理：在每一步，算法选择在分裂时给出最大信息增益的变量。然后递归地重复该过程，直到熵为零（或某个小值以解释过度拟合）。不同的算法使用不同的启发式方法来“提前停止”或“截止”以避免构建过度拟合的树。

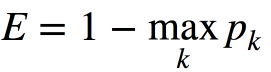


### 分类问题中拆分的其他质量标准

我们讨论了熵如何允许我们在树中形式化分区。但这只是一种启发，存在其他。

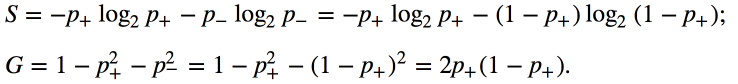


最大化该准则可以被解释为在同一子树中的同一类对象的数目对的最大化（不与基尼指数混淆）。



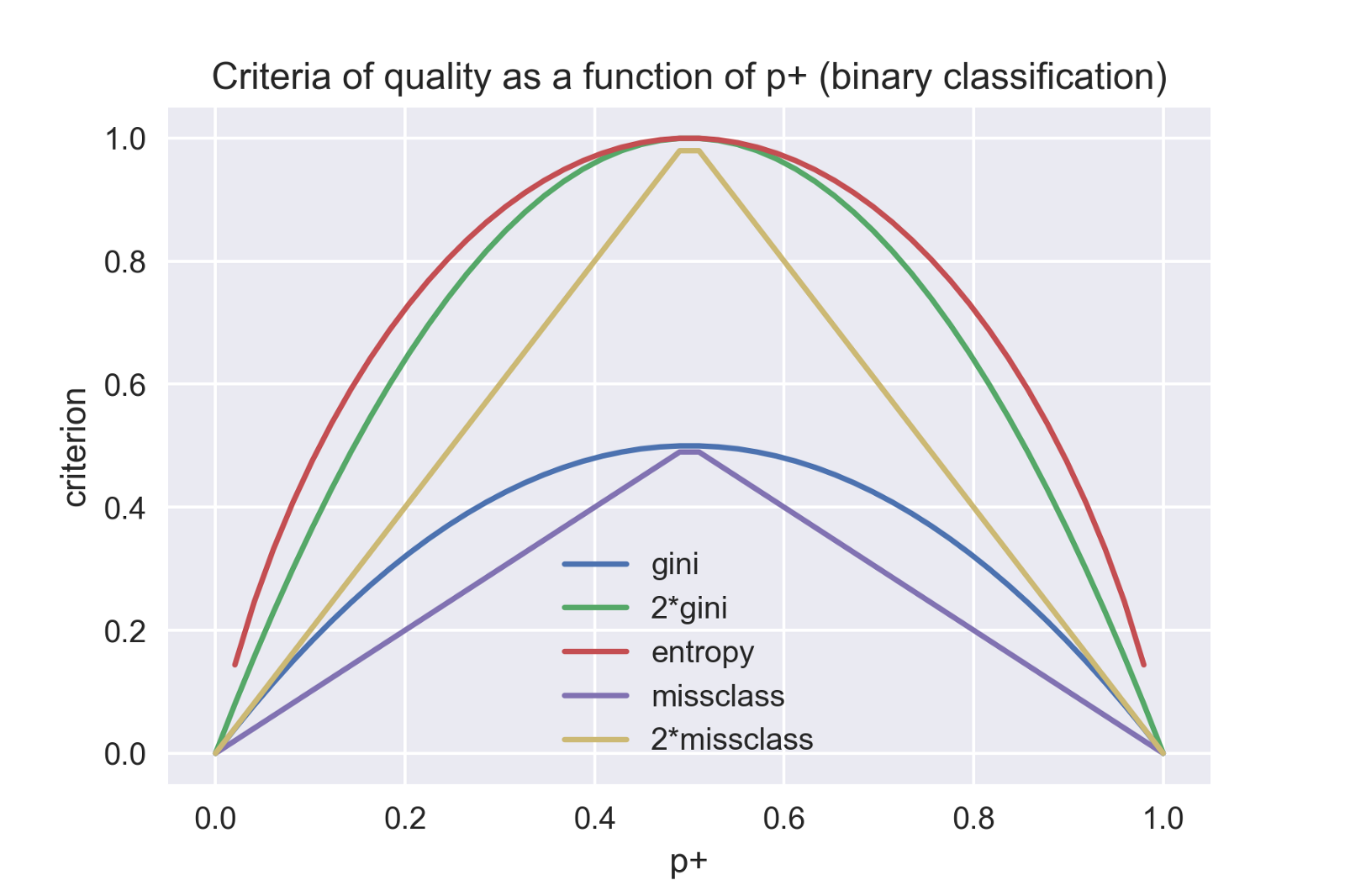
在实践中，误分类误差几乎从未被使用过，基尼不确定性和信息增益的作用类似。

对于二元分类，熵和基尼不确定度的形式如下：



其中（p+是对象具有标签+的概率）。

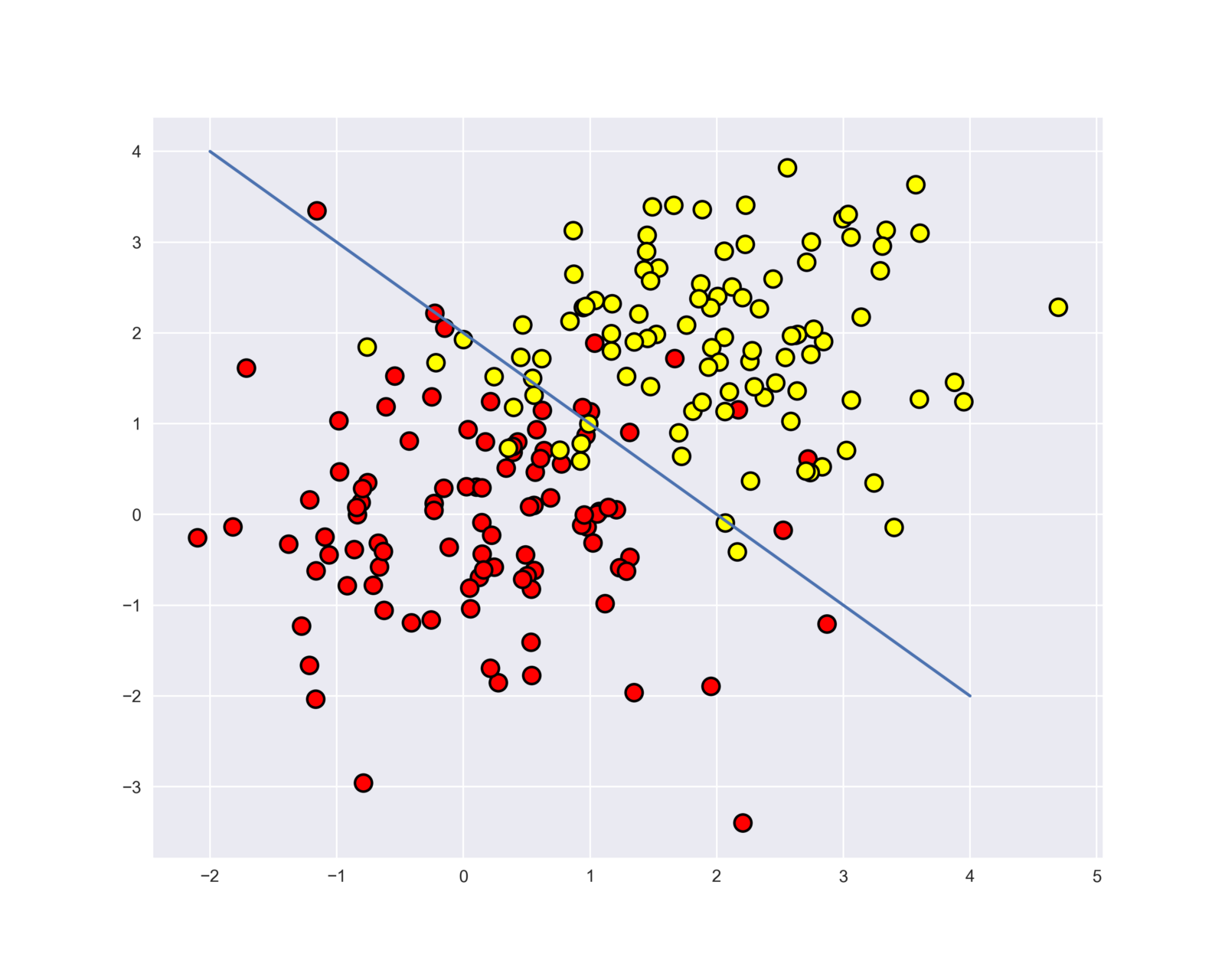
如果我们将这两个函数与参数p+对应，我们将看到熵图非常接近于基尼不确定性的图，加倍。因此，在实践中，这两个标准几乎是相同的。



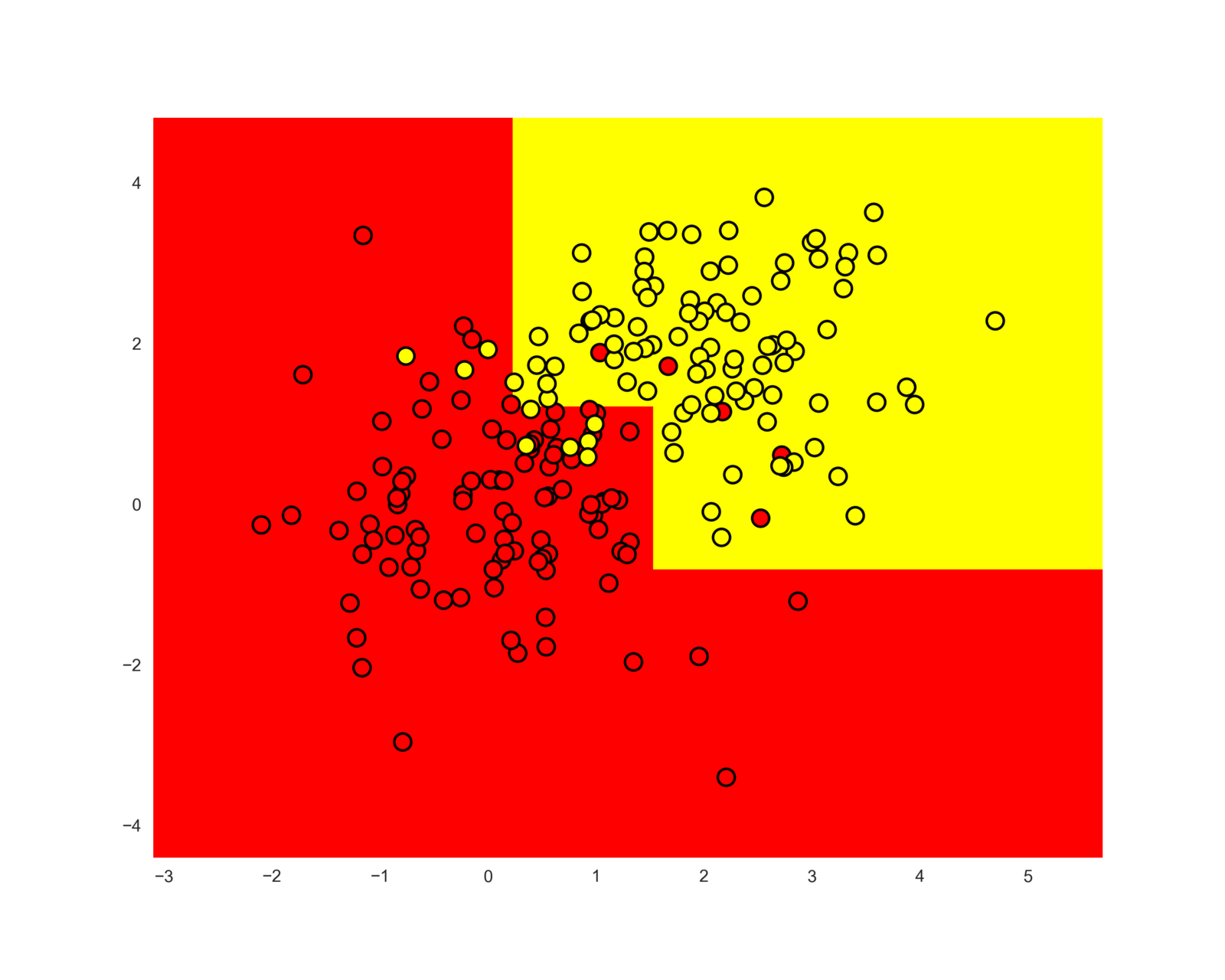
#### 例子

让我们考虑将决策树与一些合成数据进行拟合。我们将从两个类中生成样本，这两个类都是正态分布，但具有不同的平均值。

我们来绘制数据。非正式地说，这种情况下的分类问题是建立一些“好的”边界，将两个类（红点和黄点）分开。这种情况下的机器学习可以归结为选择一个良好的分离边界。直线太简单了，而每个红点的复杂曲线太复杂了，会导致我们在新样本上出错。直观地说，一些平滑的边界，或者至少是一条直线或一个超平面，对新的数据会有很好的效果。



让我们尝试通过训练Sklearn决策树来区分这两个类。我们将使用max\_depth参数来限制树的深度。让我们想象一下产生的分离边界。



这棵树看起来怎么样？我们看到树把空间“切割”成8个矩形，即树有8片叶子。在每个矩形内，树将根据其内部对象的大多数标签进行预测。

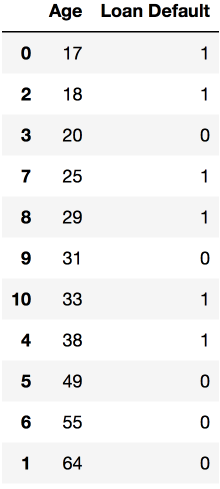
#### 我们怎么能“读”到这样一棵树？

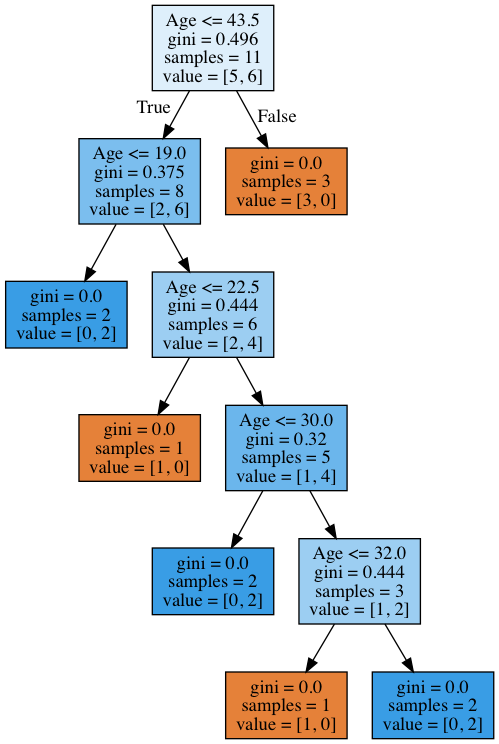
一开始，有200个样本（实例），每个类100个。初始状态的熵最大，S＝1。然后，通过比较x2与1.211的值（在上图中找到边界的这一部分），将样本分为两组。这样一来，左右两组的熵都降低了。这个过程一直持续到深度3。在这种可视化中，第一类的样本越多，顶点的橙色越暗；第二类的样本越多，蓝色越暗。开始时，两个类的样本数相等，因此树的根节点为白色。

### 决策树如何处理数字特征

假设我们有一个数字特征“Age”，它有很多唯一的值。决策树通过检查“年龄<17”、“年龄<22.87”等二元属性来寻找最佳（根据某种信息增益准则）分割。但如果年龄范围大呢？或者，如果另一个量化变量“工资”也可以在许多方面“削减”呢？在树的构造过程中，每个步骤都有太多的二进制属性可供选择。为了解决这个问题，通常使用启发式方法来限制我们比较量化变量的阈值数目。

让我们举个例子。假设我们有以下数据集：

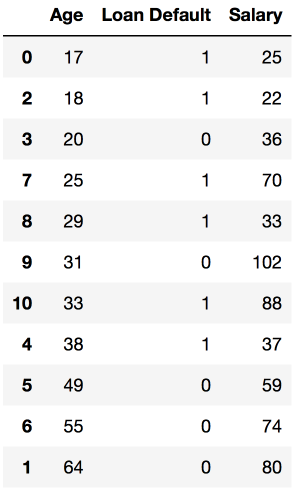




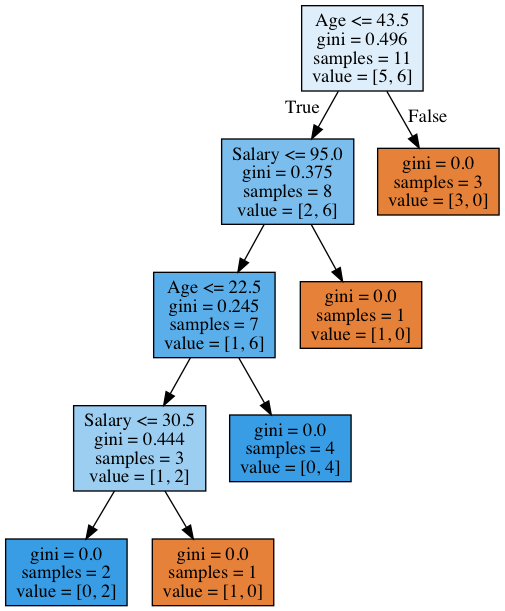
我们看到树使用了以下5个值来按年龄进行评估：43.5岁、19岁、22.5岁、30岁和32岁。如果仔细观察，这些正是目标类从1“切换”到0或从0“切换”到1的年龄之间的平均值。进一步说明，43.5是38岁和49岁的平均数；38岁的客户没有归还贷款，而49岁的客户没有归还贷款。树查找目标类切换其值的值，作为“剪切”量化变量的阈值。

既然有了这些信息，为什么你认为在这里考虑“年龄<17.5”这样的功能是没有意义的？

让我们考虑一个更复杂的例子，添加“Salary”变量（每年数千美元）。



如果我们按年龄排序，目标类（“loan default”）将切换（从1到0，反之亦然）5次。如果按工资排序，它会切换7次。树现在将如何选择功能？让我们看看。



我们看到那棵树被工资和年龄分开了。此外，特征比较的阈值为43.5岁和22.5岁，每年95k和30.5k。再次，我们看到95是88到102之间的平均值；工资为88k的人被证明是“坏”的，而102k的人是“好”的。30.5公里也是如此，也就是说，只搜索了几个按年龄和薪水比较的值。树为什么选择这些特性？因为它们提供了更好的分区（根据基尼不确定性）。

结论：处理决策树中数字特征的最简单的启发式方法是按升序对其值进行排序，并仅检查目标变量值发生变化的阈值。

此外，当数据集中有许多数字特征时，每个都具有许多独特的值，所以仅选择上面描述的阈值的前n个，即仅使用给出最大增益的Top-n。其过程是构造深度为1的树，计算熵（或基尼不确定性），并选择最佳阈值进行比较。

举例来说，如果我们除以“工资≤34.5”，左边的子组的熵为0（所有客户都是“坏的”），右边的子组的熵为0.954（3“坏的”和5“好的”，你可以自己检查，因为这将是任务的一部分）。信息增益约为0.3。如果我们除以“工资≤95”，左边的子群的熵为0.97（6“坏”和4“好”），右边的子群的熵为0（一个只包含一个对象的组）。信息增益约为0.11。如果我们以这种方式计算每个分区的信息增益，我们可以在构建一个大的树（使用所有特征）之前选择用于比较每个数字特征的阈值。

更多数字特征离散化的例子可以在类似或的帖子中找到。关于这一主题最突出的科学论文之一是“关于决策树生成中连续值属性的处理”（UM Fayyad）。KB Irani，“机器学习”，1992年）。

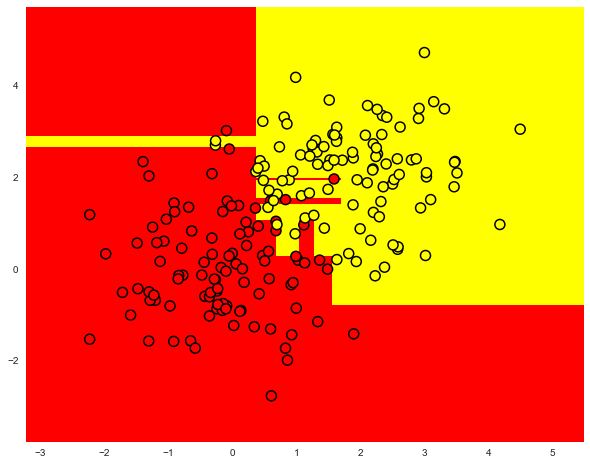
### 关键树参数

从技术上讲，您可以构建一个决策树，直到每个叶只有一个实例，但在构建单个树时，这在实践中并不常见，因为它将被过度拟合，或过于调整到训练集，并且无法很好地预测新数据的标签。在树的底部，在一定深度上，会有一些不太重要的特性的分区（例如，客户是来自Leeds还是纽约）。我们可以进一步夸大这个故事，发现四个穿着绿裤子来银行贷款的客户都没有归还贷款。即使在训练中是这样，我们也不希望我们的分类模型生成这样的特定规则。

有两个例外，树被建造到最大深度：

* 随机森林（一组树木）平均响应的树是建立在最大深度（稍后我们将讨论为什么你应该这样做）。
* 修剪树木。在这种方法中，树首先被构造到最大深度。然后，从下到上，通过比较树的质量和不使用该分区来删除树的一些节点（比较是使用交叉验证来执行的，更多内容见下文）。

下面的图片是一个例子，划分边界建立在一个过度适合的树。



处理决策树中过度拟合的最常见方法如下：

* 人工限制树叶的深度或最小样本数：树的构造只是在某个点上停止；
* 修剪这棵树。

### Scikit learn中的班级决策者

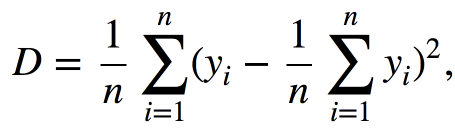
sklearn.tree.DecisionTreeClassifier类的主要参数是：

* 最大深度-树的最大深度；
* Max特征——搜索最佳分区的最大特征数量（这是需要大量特征的，因为搜索所有特征的分区将是昂贵的）；
* min\_samples\_leaf–叶中的最小样本数。此参数防止在任何叶只有少数成员的情况下创建树。

树的参数需要根据输入数据进行设置，通常是通过交叉验证来完成的，更多内容如下。

### 回归问题中的决策树

当预测一个数值变量时，树结构的思想保持不变，但是质量标准会发生变化。



其中n是叶中的样本数，Yi是目标变量的值。简单地说，通过最小化平均值周围的方差，我们寻找划分训练集的特征，这样每个叶子中的目标特征值大致相等。

#### 例子

让我们生成一些由函数分发的数据



有点噪音。然后我们将在上面训练一棵树，并展示它的预测结果。

我们知道决策树用分段常数函数逼近数据。

### 三。最近邻法

最近邻分类法（k-最近邻，或k-NN）是另一种非常流行的分类方法，有时也用于回归问题。这和决策树一样，是最容易理解的分类方法之一。潜在的直觉是你看起来像你的邻居。更正式地说，该方法遵循紧性假设：如果示例之间的距离测量得足够好，那么类似的示例更可能属于同一类。

根据最近邻法，绿球将被归为“蓝色”而不是“红色”。



例如，如果您不知道如何在在线列表中标记蓝牙耳机，则可以找到5个类似的耳机，如果其中4个被标记为“附件”，只有1个被标记为“技术”，则您还可以将其标记为“附件”。

要对测试集中的每个样本进行分类，需要按顺序执行以下操作：

1. 计算到训练集中每个样本的距离。
2. 以最小距离从训练集中选择k个样本。
3. 测试样本的类别将是k个最近邻中最常见的类别。

该方法很容易适应回归问题：在步骤3中，它返回的不是类，而是目标变量在相邻变量之间的平均值（或中值）。

这种方法的一个显著特点是其惰性 - 计算只在预测阶段进行，此时需要对测试样本进行分类。事先没有从训练实例中构造模型。相反，回想一下，在本文的前半部分中，对于决策树，树是基于训练集构建的，通过遍历树，测试用例的分类相对较快。

近邻是一种研究得很好的方法。有许多重要的定理声称，在“无尽”的数据集上，它是最佳的分类方法。经典著作《统计学习的要素》的作者认为，k-NN是一种理论上理想的算法，其使用仅受计算能力和计算量的限制。

### 近邻法在实际应用中的应用

* 在某些情况下，k-NN可以作为良好的起点（基线）；
* 在Kaggle竞赛中，k-NN通常用于元特征的构造（即k-NN预测作为其他模型的输入）或用于叠加/混合；
* 最近邻方法扩展到其他任务，如推荐系统。最初的决定可能是一个产品（或服务）的推荐，这个产品（或服务）在我们想要推荐的人的最近邻居中很受欢迎；
* 实际上，在大型数据集上，近似搜索方法通常用于最近邻。有许多开源库实现了这样的算法；请查看Spotify的库。

用k-NN进行分类/回归的质量取决于几个参数：

* 邻居的数目k。
* 样本之间的距离度量（常见的度量包括Hamming、Euclidean、cosine和Minkowski距离）。请注意，这些度量标准中的大多数要求对数据进行缩放。简单地说，我们不希望“工资”这一千差万别的特点，影响到的距离超过“年龄”，一般不到100岁。
* 邻居的权重（每个邻居可能贡献不同的权重；例如，样本越远，权重越低）。

### Scikit-learn中的kneighborsscrifier类

sklearn.NeighborsClassifier类的主要参数是：

* 权重：均匀（所有权重相等）、距离（权重与试样距离成反比）或任何其他用户定义函数；
* 算法（可选）：brute、ball\_tree、KD\_tree或auto。在第一种情况下，通过对训练集进行网格搜索来计算每个测试用例的最近邻。在第二和第三种情况下，示例之间的距离存储在树中，以加速查找最近的邻居。如果将此参数设置为“自动”，则会根据训练集自动选择查找邻居的正确方法。
* leaf\_size（可选）：如果查找邻居的算法是BallTree或KDTree，则切换到网格搜索的阈值；
* 度量标准：minkowski、manhattan、Eucliden、chebyshev或其他。

### 四。模型参数选择与交叉验证

学习算法的主要任务是能够泛化到看不见的数据。由于我们不能立即检查新的传入数据的模型性能（因为我们还不知道目标变量的真实值），因此有必要牺牲一小部分数据来检查模型的质量。

这通常是通过以下两种方式之一实现的：

* 留出数据集的一部分（保持/保持集）。我们保留一部分训练集（通常从20%到40%），在剩余数据上训练模型（原始集的60-80%），并在等待集上计算模型的性能指标（例如精度）。
* 交叉验证。这里最常见的情况是k-fold交叉验证。

在k-fold交叉验证中，模型在原始数据集（白色）的不同（k-1）子集上训练k次，并在其余子集上检查（每次都是不同的子集，如上图橙色所示）。我们得到K个模型质量评估，这些评估通常是平均的，以给出分类/回归的总体平均质量。

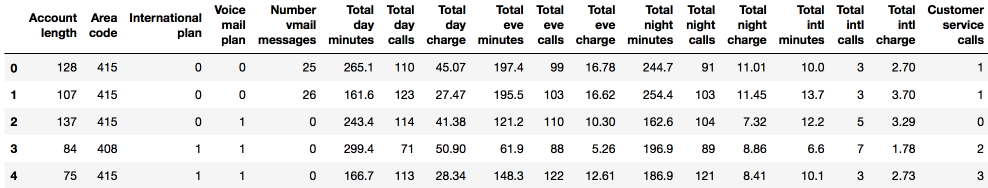
与保留集方法相比，交叉验证可以更好地评估新数据的模型质量。然而，当您拥有大量数据时，交叉验证在计算上是昂贵的。

交叉验证是机器学习中一项非常重要的技术，也可以应用于统计学和计量经济学。它有助于超参数调整、模型比较、特征评估等。更多细节可以在机器（统计）学习的任何经典教科书中找到（Sebastian Raschka的博客文章）。

### 5个。应用实例及复杂案例

### 客户流失预测中的决策树和最近邻法

让我们将数据读入一个数据帧并对其进行预处理。暂时将状态存储在单独的Series对象中，并将其从dataframe中移除。我们将训练第一个没有状态特性的模型，然后我们将看看它是否有用。



让我们分配70%的训练组（X\_train，y\_train）和30%的等待组（X\_hold out，y\_holdout）。保持集不参与模型参数的调整。我们将在调整后的最后使用它来评估结果模型的质量。让我们训练两个模型：决策树和k-NN。我们不知道哪些参数是好的，所以我们假设一些随机参数：树的深度为5，最近邻的数目等于10。

让我们用一个简单的度量来评估我们的等待集的预测质量 - 正确答案的比例（准确性）。决策树的正确回答率约为94%（决策树），而k-NN为88%（k-NN）。请注意，此性能是通过使用随机参数实现的。

现在，让我们使用交叉验证来标识树的参数。我们将调整每个分割中使用的最大深度和最大特征数量。GridSearchCV的工作原理如下：对于max\_depth和max\_features的每个唯一值对，使用5倍交叉验证计算模型性能，然后选择最佳参数组合。

让我们列出交叉验证的最佳参数和相应的平均精度。

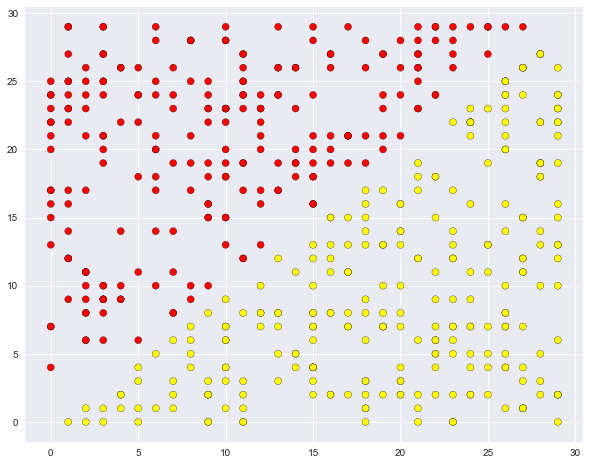
让我们画出结果树。由于它不完全是一个玩具示例（它的最大深度是6），图片不是那么小，但是如果您在本地打开从课程回购中下载的相应图片，您可以在树上“行走”。

现在，让我们调整k-NN的邻居k的数量：

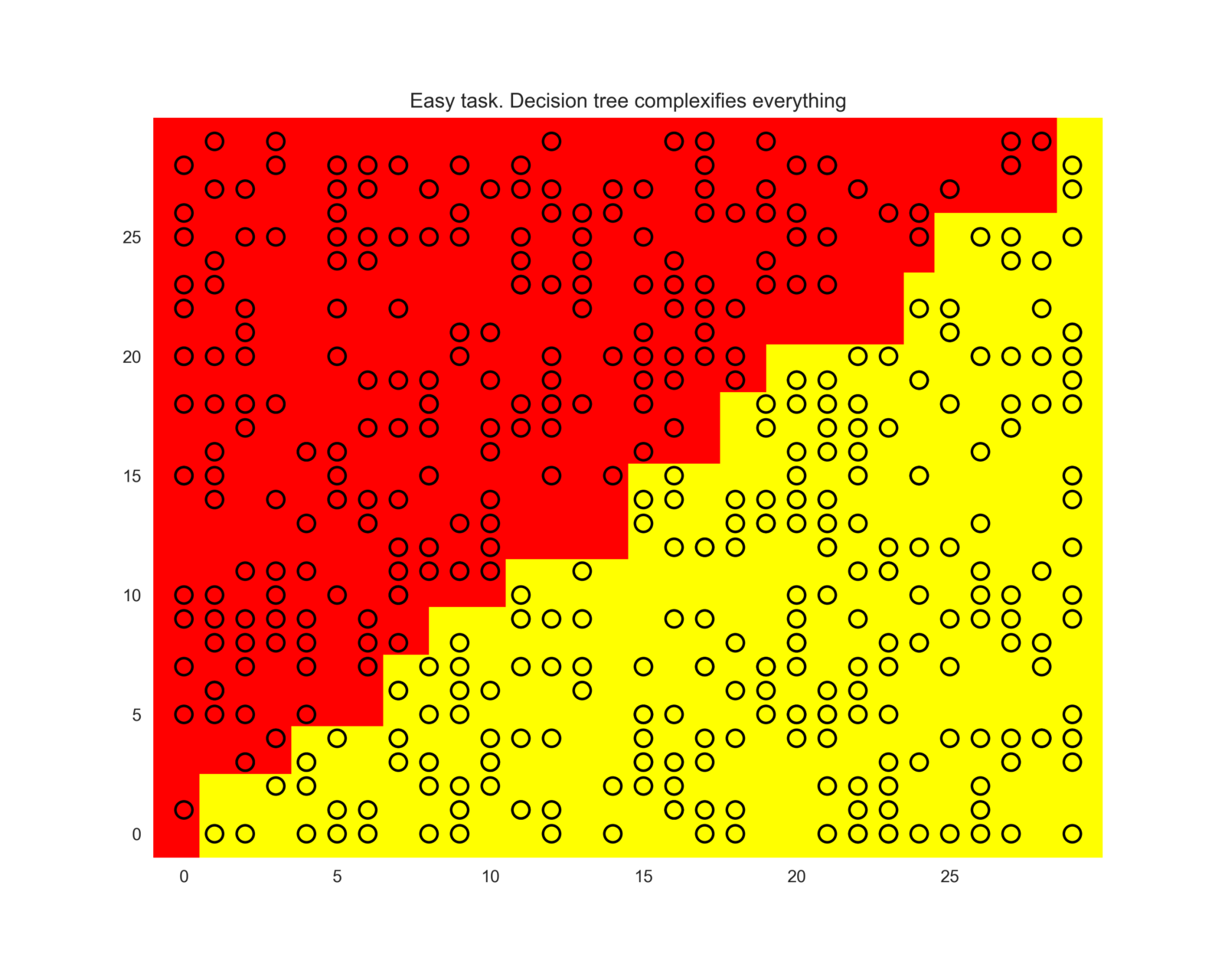
在这里，证明了该树优于最近邻算法：交叉验证的准确率为94.2%/94.6%，并且保持良好。决策树的性能非常好，在这个例子中，即使是随机森林（现在我们把它看作是一堆更好地协同工作的树）也不能获得更好的性能（95.1%/95.3%），尽管经过更长时间的训练。

### 决策树的复杂情况

为了继续讨论这些方法的优缺点，让我们考虑一个简单的分类任务，在这个任务中，一个树可以很好地执行，但是它以一种“过于复杂”的方式执行。让我们在平面上创建一组点（2个特征），每个点都是两个类中的一个（红色为1，黄色为-1）如果你把它看作一个分类问题，它看起来很简单：类之间用一行隔开。

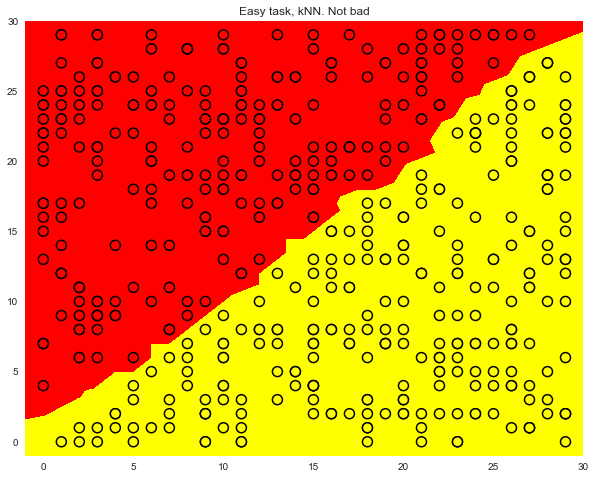


然而，决策树构建的边界太复杂，加上树本身很深。另外，想象一下树将如何糟糕地概括到30 x 30平方以外的空间，该空间构成了训练集。



我们得到了这个过于复杂的构造，尽管解只是一条直线x1=x2。

最近邻法优于树法，但仍不如线性分类器（our）。

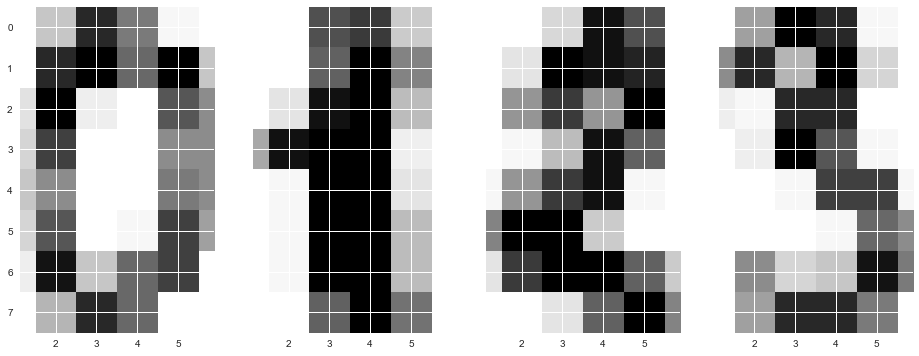


### 决策树与k-NN在MNIST手写数字识别中的应用

现在让我们来看看这两个算法在实际任务中是如何执行的。我们将在手写数字上使用sklearn内置数据集。这个任务是一个例子，其中k-NN工作得出奇的好。

这里的图片是8x8矩阵（每个像素的白色强度）。然后，每一个这样的矩阵被“展开”成一个长度为64的向量，我们得到一个对象的特征描述。

让我们画一些手写数字。我们看到它们是可分辨的。



接下来，让我们做与上一个任务相同的实验，但是，这次，让我们更改可调参数的范围。

让我们选择70%的数据集进行训练（X\_-train，y\_-train）和30%的数据集进行训练（X\_-holdout，y\_-holdout）。holdout集不会参与模型参数调整；我们将在最后使用它来检查结果模型的质量。

让我们用随机参数训练一个决策树和k-NN，并对保持集进行预测。我们可以看到k-NN做得更好，但注意这是随机参数。

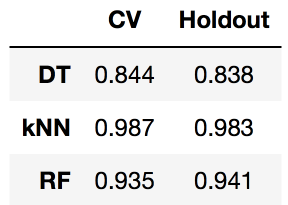
现在让我们像以前一样使用交叉验证来优化我们的模型参数，但是现在我们将考虑到我们有比上一个任务中更多的特性：64。

让我们看看最佳参数组合和交叉验证的相应精度：

已经超过66%，但还不到97%。k-NN在这个数据集上工作得更好。对于一个最近的邻居，我们能够在交叉验证中达到99%的猜测。

让我们在同一个数据集上训练一个随机林，它在大多数数据集上比k-NN更有效。但我们这里有个例外。

您可以正确地指出，我们在这里没有调整任何RandomForestClassifier参数。即使有调谐，训练精度也不能达到98%，就像最近邻的情况一样。



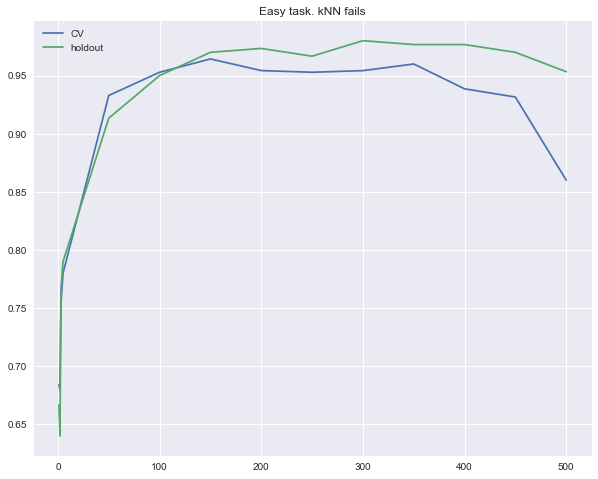
这个实验的结论（和一般建议）：首先检查数据上的简单模型：决策树和最近邻（下一次我们还将向这个列表中添加logistic回归）。也许这些方法已经足够有效了。

### 近邻法的复情形

让我们考虑另一个简单的例子。在分类问题中，其中一个特征将与响应向量成正比，但这对最近邻方法没有帮助。

一如既往，我们将研究交叉验证和保持集的准确性。在最近邻法中，我们构造曲线来反映这些量对n\_邻域参数的依赖性。这些曲线称为验证曲线。

可以看出，即使在很宽的范围内改变最近邻的数目，具有欧氏距离的k-NN也不能很好地解决这个问题。



相反，决策树很容易“检测”隐藏的依赖关系的数据，尽管限制最大深度。

在第二个例子中，当k-NN遇到困难时，树完美地解决了问题。然而，使用欧几里得距离比使用该方法更为不利。它不允许我们揭示一个特性比其他特性好得多。

### 6。决策树与最近邻法的优缺点

#### 决策树的优缺点

赞成的意见：

* 生成清晰的人类可理解的分类规则，例如“如果年龄<25岁且对摩托车感兴趣，则拒绝贷款”。此属性称为模型的可解释性。
* 决策树可以很容易地可视化，即模型本身（树）和某个测试对象（树中的路径）的预测都可以“解释”。
* 快速训练和预测。
* 少量模型参数。
* 支持数字和分类特征。

欺骗：

* 这些树对输入数据中的噪声非常敏感；如果对训练集稍加修改（例如删除一个特征，添加一些对象），整个模型可能会发生变化。这削弱了模型的可解释性。
* 决策树建立的边界分离方法有其局限性，它由垂直于某一坐标轴的超平面构成，在实际应用中质量不如其他方法。
* 我们需要避免通过修剪过度拟合，在每个叶中设置最小数量的样本，或者为树定义最大深度。请注意，对于所有机器学习方法，过度拟合都是一个问题。
* 不稳定。对数据的微小更改可以显著更改决策树。这个问题用决策树集合来解决（下次讨论）。
* 最优决策树搜索问题是NP完全问题。一些启发式算法在实践中被使用，例如贪婪地搜索具有最大信息增益的特征，但是它不能保证找到全局最优树。
* 难以支持数据中缺少的值。Friedman估计，大约50%的代码支持CART中的数据间隙（该算法的改进版本在sklearn中实现）。
* 该模型只能进行插值而不能进行外推（对于随机森林和树木生长也是如此）。也就是说，决策树对特征空间中超出训练集所设置的边界框的对象进行持续的预测。在我们使用黄色和蓝色球的例子中，这意味着模型对所有位置大于19或小于0的球给出了相同的预测。

#### 近邻法的优缺点

赞成的意见：

* 简单的实现。
* 学得很好。
* 通常，该方法不仅是分类或回归的好的第一解决方案，而且也是建议。
* 它可以通过选择合适的度量或核来适应特定的问题（简而言之，核可以在保持k-NN方法不变的情况下，为图等复杂对象设置相似操作）。顺便说一句，一个前排名前1的kaggler喜欢最简单的k-NN，但是有调整过的对象相似性度量。
* 很好的解释能力。但也有例外：如果邻居的数量很大，可解释性就会下降（“我们没有给他贷款，因为他与350个客户相似，其中70个是坏客户，比数据集的平均值高出12%）。

欺骗：

* 与算法组合相比，该方法被认为是快速的，但在现实生活中，用于分类的邻域数目通常很大（100-150），在这种情况下，该算法的运算速度不如决策树。
* 如果一个数据集有许多变量，很难找到正确的权重，也很难确定哪些特征对分类/回归不重要。
* 依赖于对象之间选定的距离度量。默认情况下选择欧几里德距离通常是没有根据的。通过对参数进行网格搜索可以找到一个很好的解决方案，但对于大型数据集来说，这将非常耗时。
* 没有理论上的方法来选择邻域的数目 - 只有网格搜索（尽管这通常适用于所有模型的所有超参数）。在邻域数目较少的情况下，该方法对异常值比较敏感，即倾向于过拟合。
* 通常情况下，由于“维度诅咒”的原因，当有很多特性时，它不能很好地工作。ML社区的知名成员Pedro Domingos教授在他的流行论文《关于机器学习的一些有用的知识》中谈到了这一点；在《深度学习》一书中也描述了“维度的诅咒”。

这是很多信息，但是，希望这篇文章能在很长一段时间内成为您的参考：）

### 7号。作业#3

课程每周（2018年10月1日）都会公布作业的完整版本。同时，您可以使用演示版本：。

### 8个。有用资源

* 决策树和k个最近的邻居几乎都包含在每本ML书中。我们推荐“模式识别和机器学习”（C.Bishop）和“机器学习：概率视角”（K.Murphy）。
* 《机器学习在行动》（P.Harrington）一书将带您了解纯Python中经典ML算法的实现。
* [Scikit-learn](http://scikit-learn.org/stable/documentation.html) library. These guys work hard on writing really clear documentation.
* Scipy 2017 [scikit-learn tutorial](https://github.com/amueller/scipy-2017-sklearn) by Alex Gramfort and Andreas Mueller.
* One more [ML course](https://github.com/diefimov/MTH594_MachineLearning) with very good materials.
* [Implementations](https://github.com/rushter/MLAlgorithms) of many ML algorithms. Good to search for decision trees and k-NN.
* And many others, feel free to share in comments.

Author: [Yury Kashnitskiy](https://www.linkedin.com/in/festline/). Translated and edited by [Christina Butsko](https://www.linkedin.com/in/christinabutsko/), Gleb Filatov, and [Yuanyuan Pao](https://www.linkedin.com/in/yuanyuanpao/).